International Journal of Engineering & Technology, 7 (3.24) (2018) 651-654

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Implementation of Deep Learning based Rendering De-Noising Accelerator

[1]SeongHyeon Han, [*2]KwangYeob Lee

[1]Department of Computer Engineering, Seokyeong University, 124, Seogyeong-ro, Seongbuk-gu, Seoul, 02713,Republic of Korea
*[*2]Department of Computer Engineering, Seokyeong University, 124, Seogyeong-ro, Seongbuk-gu, Seoul, 02713, Republic of Korea*
**Corresponding author Email: [*]kylee@skuniv.ac.kr*

Abstract

Background/Objectives: In this paper, we implemented the neural network of learning based filtering algorithm, which eliminates noise generated by ray-tracing, through Verilog HDL.
Methods/Statistical analysis: The neurons used in the learning based filtering algorithm are divided into five stages: IDLE, SIGN, MUL, SUM, and ACT. Each stage is processed in one cycle through the FSM(Finite State Machine).These neurons were organized into a number of layers. The operation used fixed point.
Findings: The neural network has a large amount of computation, but since the computation is simple, it can be processed quickly by hardware implementation. It has a hidden layer (10 neurons with 36 inputs), an output layer (6 neurons with 10 inputs), each layer has 5 stages, so you can get the filtering parameters after 10 cycles.
Improvements/Applications: Verilog HDL can be synthesized and downloaded to the FPGA to operate up to 185MHz.

Keywords: *Ray tracing, Rendering noise, Leaning based filtering, Neural network, FPGA, MLP*

1. Introduction

With the recent development of GPU, the graphics application field is rapidly expanding. Among them, the ray tracing algorithm is a rendering algorithm that produces high-quality images. Ray tracing is used in many fields such as games and 3D animation, and application fields are expanding according to the development of displays and VR. The ray tracing algorithm is an algorithm that determines the pixel color of an image by emitting a ray of light from a light source, and tracing the path of the ray which is reflected or refracted back to the light source. Assuming that an image is created through a white ray on a black image, as shown in Figure 1, a small number of rays will degrade the image and noise will occur, and a large number of rays will lead to a higher image quality, as shown in Figure 2 [1].

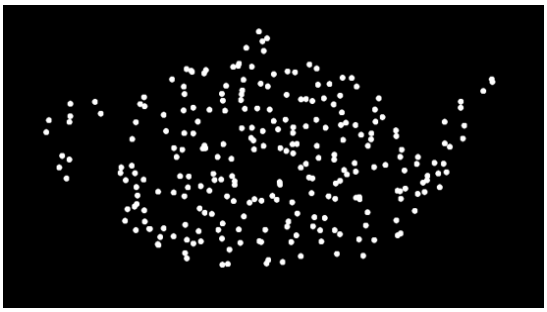


Figure 1:. Image with a small number of rays

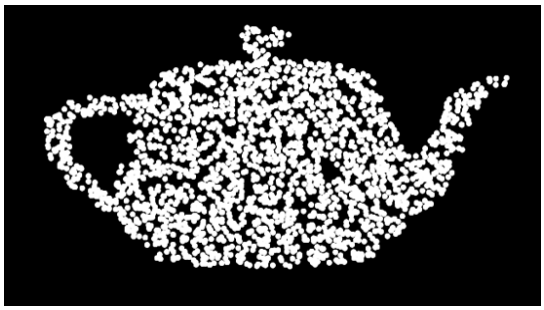


Figure 2:. Image with a large number of rays

Figure 3:.8 rays/pixel image (157 secs)

Figure 4:.32 rays/pixel image (710 secs)

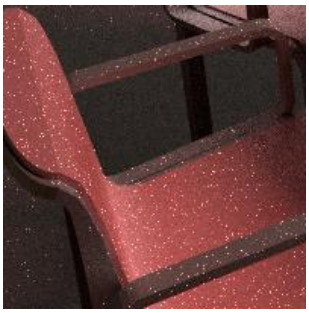



Figure 5:.256 rays/pixel image (3h 48m)

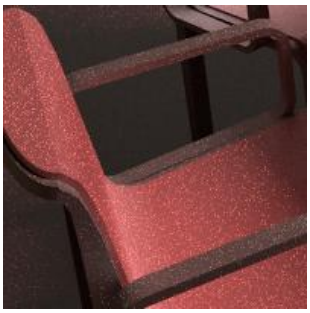



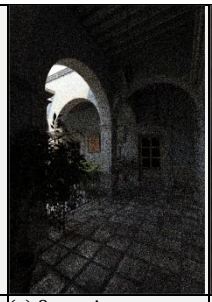
Figure 6:.2,048 rays/pixel image (12h 48m)

However, rendering with a large number of rays to improve the quality of the image requires a large amount of operation. Figure 3~6 shows the rendering result of a 1,024 x 1,024 resolution image with a different number of rays. Figure 3 shows the results of rendering with 8rays/pixel. The operation took 157 seconds, but noise occurred in the image. Figure 4 and Figure 5 show the rendering results with an increased number of rays, and show that the noise gradually disappeared. Figure 6 shows rendering results with 2,048 rays. Although the image quality has improved, the operation took more than 12 hours.

Using a lot of rays for a high-quality image takes a long time to operate. To solve this problem, an algorithm has been introduced that removes the noise from the image through filtering operations. These algorithms include, NLM which removes noise using Non-Local Means filtering [2], RPF which computes and filters the functional dependency between the noise image and noise-free image [3], SURE which removes noise using SURE (Stein's Unbiased Risk Estimator) based filter [4], RD which combines the color buffer and feature buffer [5], and LBF which filters the dependency between the feature and color through a neural network [6].

Table 1 shows an image with the NLM, RPF, SURE, RD, and LBF algorithms applied to an image rendered with 8 rays [6].Structural SIMilarity (SSIM) is an algorithm for measuring the similarity between two images, and the closer to 1, the more similar the two images are. Each filtering algorithm is the result of SSIM calculation compared with a high-quality image rendered with 64K rays [7].

Table 1: Ray tracing noise filtering algorithms

image						
	(a) 8 rays image	(b) NLM	(c) RPF	(d) SURE	(e) RD	(f) LBF
Time	125 secs	147 secs	815 secs	299 secs	161 secs	133 secs
SSIM	0.458	0.754	0.892	0.841	0.883	0.889

2. Leaning Based Filtering

Among the filtering algorithms used for removing rendered noise, LBF (Learning Based Filtering) not only produces high-quality images, but also has an advantage with operation speed using CUDA, a GPGPU technology [6].

LBF is basically a filtering algorithm, so it is a post-processing. LBF creates the final image through rendering, feature extractor, neural network forward pass, and filtering.

The rendering process uses a small number of rays to create an image with noise. Usually, 8 or 16 rays are used for rendering. If more rays are used, the rendering takes a long time. PBRT is used for rendering [8]. The PBRT renderer operates on multi-core using OpenMP. Images rendered with a small number of rays generate noise, as shown in Table 1 (a).

The feature extractor process extracts various features used in the rendering. The extracted features are called primary features and are shown in Table 2. There are 18 primary features in total.

Table 2:. Primary Features

	Image position X, Y	Colors R, G, B	World Position X, Y, Z	Normals X, Y, Z	Textures X, Y, Z	Visibility
Number of features	2	3	3	3	6	1

In this process, the extracted primary features are converted into secondary features. There are a total of 36 secondary features, as shown in Table 3.

Table 3:. Secondary Features

	Feature statistics	MAD	Gradient	MD	SPP
Number of features	20	5	5	5	1

In the neural network forward pass process, the secondary features obtained from the previous process is used as an input for the neural network. The structure of the neural network is a multi-layered perceptron structure with three layers. The input layer has 36 neurons, the hidden layer has 10 neurons, and the output layer has 6 neurons as shown in figure 7.

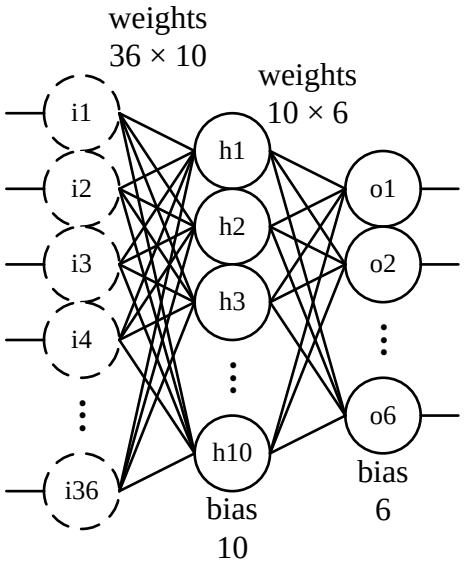


Figure 7: Architecture of LBF neural network

For the activation functions used in LBF, sigmoid was used for the hidden layer and ReLU [7] was used for the output layer. The 6 output values obtained from the output layer are the filter parameters used for filtering.

Figures 8 and 9 show the result of rendering through PBRT and the LBF image. The image resolution is 1,280 × 800, and the image in Figure 8 used 16 rays, and the image in Figure 9 used LBF.

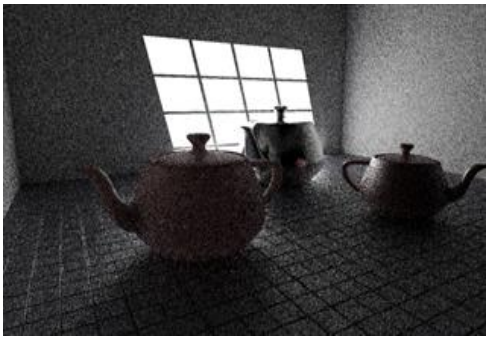


Figure 8: 16 rays rendering image

Figure 9: LBF image

The experiment used 4 Cores 8 Threads 3.4Ghz Intel CPU with hyper-threading, 24GB DDR3 main memory and Nvidia GTX 1070 GPU as shown in table 4.Rendering operates with 8 threads, filtering uses 16 × 1024 blocks, and 128 threads were used per block.

Table 4: Processing time

	Processing time
Rendering	29 secs
Feature Extraction	4.42 secs
Neural network forward pass	0.65 secs
Filtering	1.73 secs
Total time	35.8 secs

3. Learning based Filtering Accelerator

The conventional LBF algorithm uses GP-GPU to process the operation. In this thesis, a neural network accelerator was designed which operates as an FPGA through Verilog HDL to increase the operation speed.

Neurons in the input layer were not designed since their purpose is just for a simple data transfer. Two kinds of neurons were designed because the neurons in the hidden layer have 36 inputs and the neurons in the output layer have 10 inputs. In other words, the neurons used in the hidden layer instantiated 36 neurons with 36 inputs, and the neurons used in the output layer instantiated 10 neurons with 10 inputs.

Neurons in each layer operate in parallel to increase the operation speed, but each layer must wait until all of the neurons complete the operation. Therefore, the output layer must wait when the hidden layer operates, and the hidden layer must wait when the output layer operates.

The neuron of the hidden layer has 1,152-bit input data and enable signal for input, and 32-bit output data and done signal for output. The reason for 1,152-bit input is because it receives 36 inputs and 32-bit fixed point numbers as input. Accordingly, the neuron of the output layer with 10 inputs has 320-bit input data, and the other input and output ports are the same as the hidden layer neuron.

Each neuron has five states: IDLE, SIGN, MUL, SUM, and ACT. The IDLE state is the state before the data enters the operation of the corresponding layer, or when the layer is in a standby state. In this state, data from the input port is cut into 32 bits and stored in the internal register, and the intermediate values used internally are initialized to 0. In the SIGN state, the sign is determined by performing an XOR operation on the most significant bit of the weight stored as internal local parameters and the input data. If the most significant bits are the same, the most significant bit of the intermediate storage value is stored as 0 to take a positive number, and if they are different it is stored as 1. In the MUL state, the input data is multiplied by the weight. Since the sign bit was determined in the SIGN state, 31 bits excluding the sign bit are multiplied. By using fixed point numbers, it was designed to operate in one clock cycle. In the SUM state, since the sign was determined in the SIGN state, the positive and negative numbers are divided and added accordingly. The sum of the positive numbers and the negative numbers are added up and compared. If the sum of the positive numbers is larger, the difference between the sum of the positive and negative numbers is found, and the sign bit, which is the most significant bit, is changed to 0 and stored as a positive number. If the sum of the negative numbers is larger, the difference between the sum of the positive and negative numbers is found, and the most significant bit is changed to 1 and stored as a negative number. In the ACT state, if the value is a positive number, the corresponding value is sent to the output value, and if it is a negative number, the value is changed to 0 and then sent to the output value. Figure 10 shows each state of the neuron in cycles.

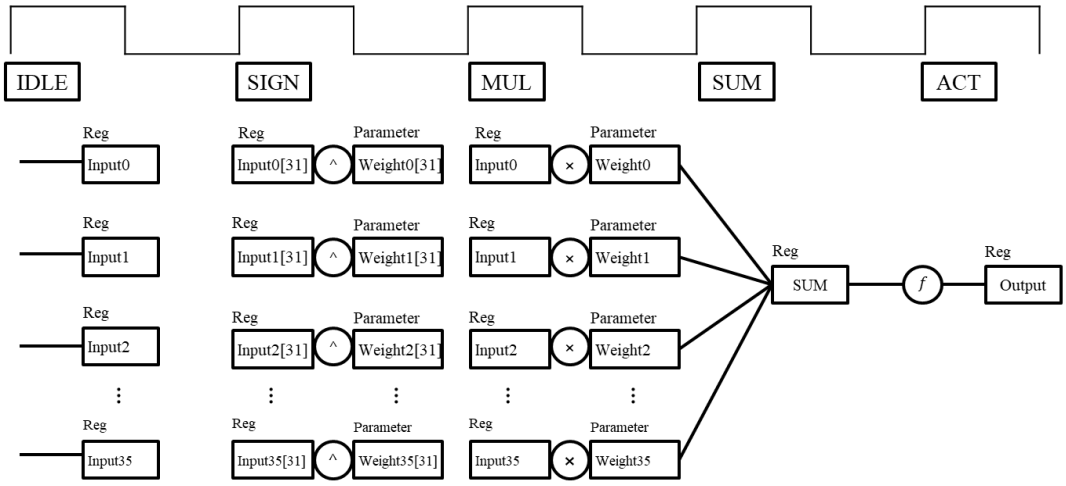


Figure 10:.FSM of Neuron

The neural network used in the LBF has weight and bias in the hidden and output layer. Accordingly, there are 360 weights between the input and hidden layer, and 60 weights between the hidden and output layer. The weight and bias were stored as local parameters.

When the neural network was integrated with the Xilinx VCU-108 Board, the maximum operating frequency was 187 MHz, the setup time was 0.54 ns, the hold time was 0.12 ns, and 1,411 registers and 4,322 LUT (Look Up Table) were used internally. For stability, an async module was applied between the clock and the network, and the total operating frequency of the FPGA was 200MHz, and the network was operated at 180 MHz. Since the clock cycle of 180MHz is 5.56 ns, 5 cycles are required for each layer, and since it has 2 layers, it is possible to find the parameters required for the filter once every $5 \times 2 \times 5.56$, 56 ns.

The following Table 5 compares the resource usage with neural network accelerators through other FPGAs. For comparison, the relevant network structures are listed together.

Table 5:. Result of synthesis

	[9]	[10]	LBF accelerator
Neural network architecture (input/hidden/output)	36/10/10	25/32/5	36/10/6
Register	3,050	3,450	1,411
LUTs	3,100	3,283	4,322

4. Conclusion

In this thesis, the neural network used in the LBF algorithm for removing ray tracing rendering noise, was implemented in FPGA through Verilog HDL. Each layer was operated sequentially, and each neuron was operated in parallel to increase the operation speed. Since the operating frequency of the neural network was 180 MHz and 10 cycles were required to complete the operation, the filter parameters could be obtained once every 56 ns.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP). (No. 2016-0-00204, Development of mobile GPU hardware for photo-realistic real time virtual reality)and supported by Seokyeong University in 2017

References

[1] Introduction to Ray Tracing: a Simple Method for Creating 3D Images Retrieved fromhttps://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/raytracing-algorithm-in-a-nutshell

[2] Sen, P., & Darabi, S. (2012). On filtering the noise from the random parameters in Monte Carlo rendering. ACM Trans. Graph., 31(3), 18-1.

[3] Van De Ville, D., & Kocher, M. (2009). SURE-based non-local means. IEEE Signal Processing Letters, 16(11), 973-976.

[4] Rousselle, F., Manzi, M., &Zwicker, M. (2013, October). Robust denoising using feature and color information. In Computer Graphics Forum (Vol. 32, No. 7, pp. 121-130).

[5] Kalantari, N. K., Bako, S., & Sen, P. (2015). A machine learning approach for filtering Monte Carlo noise. ACM Trans. Graph., 34(4), 122-1.

[6] Pharr, M., Jakob, W., & Humphreys, G. (2016). Physically based rendering: From theory to implementation. Morgan Kaufmann.

[7] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.

[8] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[9] Raharjo, P. M., Rif'an, M., &Sulistyanto, N. (2012). The Implementation of Feedforward Backpropagation Algorithm for Digit Handwritten Recognition in a Xilinx Spartan-3. Jurnal EECCIS, 4(2), 17-21.

[10] Sejinchoi, (2017), Implementation of GP-GPU based deep learning parallel algotithm and neural network accelerator. (Masters dissertation). Seokyeong University, Seoul, Republic of Korea Retrieved from http://www.riss.kr/link?id=T14793399